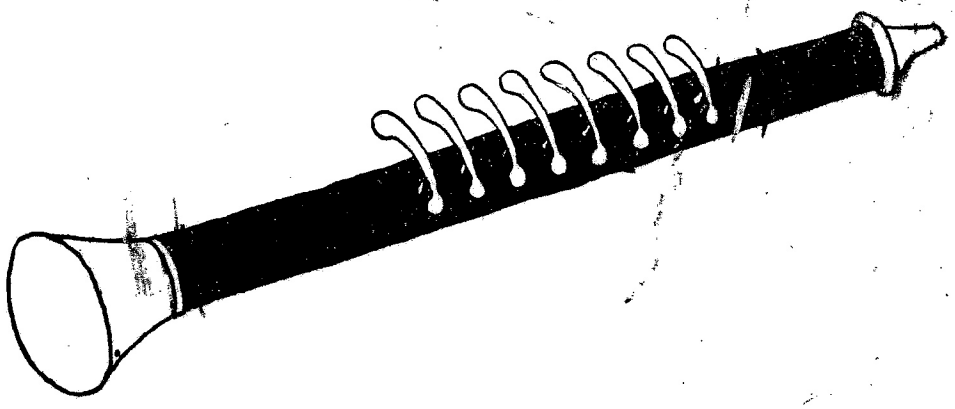
Microcomputer

PHYSICS LAB



SOUND

CROSS
EDUCATIONAL SOFTWARE

TABLE OF CONTENTS

INTRODUCTION

This is a kit of parts for doing sound experiments with an Apple computer. The easiest experiment is the Audio Oscillator experiment since you need only assemble a speaker. The microphone has a few more parts and is harder to assemble. Both circuits can be assembled directly on 16 pin sockets. For rough use they can be put inside a project box and connected to the game socket with a ribbon cable.

You do not need the interface to do part of the Audio Oscillator program. It uses the original Apple speaker. You can try it now, before you've assembled an interface.

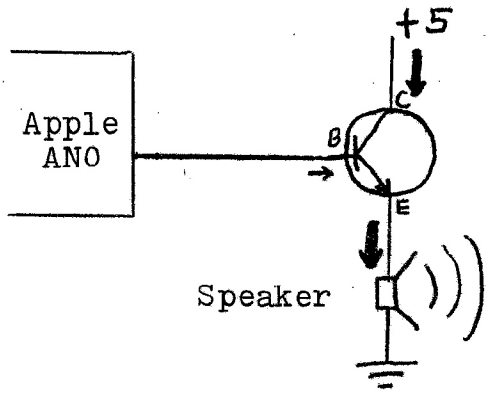
KIT CONTENTS:

PROGRAM DISKETTE
MANUAL
SPEAKER
MICROPHONE (RADIO SHACK #270-092A)
THREE NPN TRANSISTORS (2N3904 or 2N2222)
TWO 16 PIN IC HEADER PLUGS
POTENTIOMETER
10 MICROFARAD CAPACITOR
3 AND 4 WIRE CABLES

SPEAKER

HOW IT WORKS

A loudspeaker moves and makes sound when current passes through it. Apple's game paddle connector can supply a little current at its annunciator outputs, but not enough to drive a speaker. One transistor must amplify the current as shown in the diagram. The annunciator sends current into the base, which causes more current to flow down through the collector. Both currents pass out through the emitter and the speaker down to ground.



Speakers must be turned on and off very quickly to make a high frequency audible sound. This line in Applesoft makes the internal speaker buzz:

```
FOR I = 1 TO 100: X = PEEK (-16336): NEXT I
```

BASIC is too slow. At top speed it can turn the speaker on and off fast enough to make a low pitched buzz.

Many different types of tone subroutines can be written in machine language. This disk of sound programs uses tone subroutines with a very precise delay between motions of the speaker. This makes it possible to select the frequency with a typical error of 1% over most of the audio range. Here is a simple way to call the machine language tone subroutine:

```
10 PRINT CHR$(4); "BLOAD TWO SPEAKER TONES"
20 INPUT "DELAY = ";D
30 POKE 1, D / 256
40 POKE 0, D - 256 * PEEK(1)
50 CALL 768
```

where D = the delay time, from 0 to 32767.
location 0 is the low byte of the delay
location 1 is the high byte of the delay
Return from subroutine after any key is pressed.

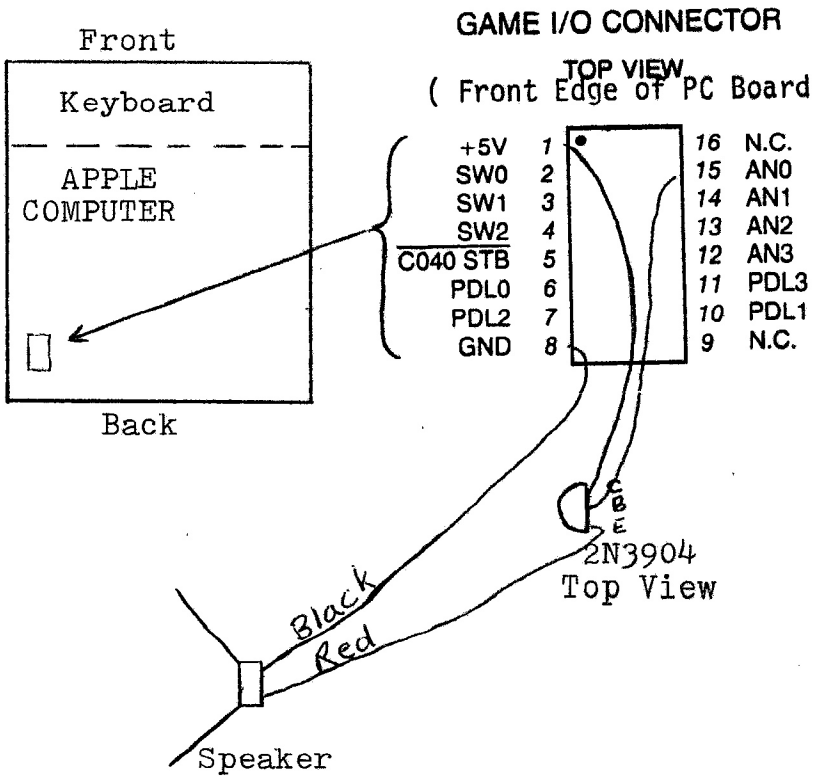
ASSEMBLING THE SPEAKER

We have included one speaker with this kit for doing resonance experiments. We recommend using two wires of the four conductor cable to connect just one speaker at first. Another speaker can be added at any time.

The game connector is a 16 pin socket at the right rear of the Apple. Pin 1 is closest to the keyboard. You will make connections to pin 1 (5 volts), pin 8 (ground) and pin 15 (annunciator 0).

On the 16 pin header plug in this kit, attach the collector of a transistor to 5 volts (pin 1). Attach the base to AN0 (pin 15). Using a long piece of cable, connect the red wire to the emitter of the transistor and the black to ground (pin 8). The other end of the cable goes to the speaker. Connect red and black to the two speaker terminals. The speaker is now finished and ready for the audio oscillator program.

We have included only one speaker with this kit, but there is a spare transistor and the computer program will support more speakers. If you add a second speaker, use the same procedure with AN1 (pin 14), another transistor, and the yellow and green wires on the four conductor cable. Two speakers might be needed to demonstrate audio beats between close frequencies, but beats can also be heard between an external speaker and Apple's original speaker.



276-1603 2N3904

NPN TRANSISTOR

This 276-1603 is designed as a general purpose amplifier and switch. The useful dynamic range extends to 100 mA as a switch and to 100 MHz as an amplifier. For complementary use with PNP 276-1604 (Type 2N3906)

Collector-Base Voltage 60v

Collector-Emitter Votlage 40v
(Between 10uA and 200uA, collector current when the base-emitter diode is open-circuited.)

Emitter-Base Voltage 6v

Pin 1 Emitter
Pin 2 Base
Pin 3 Collector

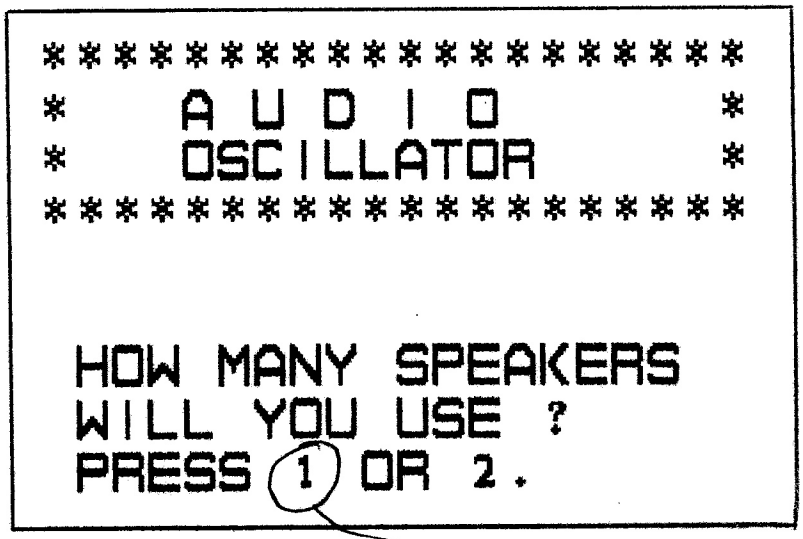
BOTTOM VIEW

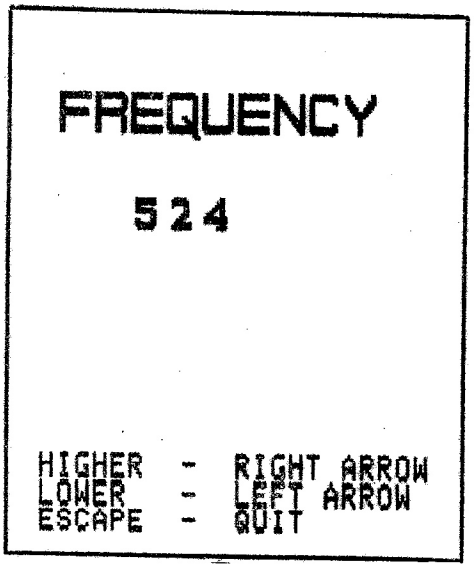
AUDIO OSCILLATOR

This program can do two physics experiments. First, it can demonstrate beats between two speakers. These can be two external speakers or else Apple's internal speaker and the speaker from this kit. All you have to do is run the audio oscillator program and select the two speaker option at the beginning. The program will ask which two speakers you are using. Then it will ask for two frequencies for the speakers. For example, frequencies of 400 and 410 hertz will make obvious beats. Since the precision of this program is only about 1% you might find that frequencies of 400 and 402 hertz are identical and don't make beats. The frequencies must be more than 1% apart.

Second, the audio oscillator can find resonances in air cavities. Use any hollow object such as a cardboard tube, drinking glass, or bottle. Place the speaker at one end of the resonator. Run the oscillator program and chose just one speaker. Use the right and left arrow keys to raise and lower the frequency. If you hold down an arrow the tone will change smoothly and it will be easier to hear the resonance. When you let up the arrow key the tone will be steady and the screen will update the frequency after a second or two.

The error in the frequency is about 1 to 5 %. This is better than the error in hearing a resonance. The error can be reduced by more careful calibration and adjustment of the constants in line 300 of the audio oscillator program.





AUDIO OSCILLATOR PROGRAM

```
LIST

1  GOTO 100: REM  AUDIO OSCILLATOR PROGRAM
2  REM  COPYRIGHT 1983 DR. MARK CROSS

100  HIMEM: 31744: LOMEM: 24700: GOSUB 63000: REM   SET UP THE GRAPE HI-RES PRINTER
110  PRINT  CHR$ (13); CHR$ (4);"BLOAD TWO SPEAKER TONES"
111  PRINT  CHR$ (4);"BLOAD REPEAT TONES,A$6000": REM  USED FOR ONE SPEAKER WITH ARROW KEYS TO CHANGE THE FREQUENCY
115  ONERR  GOTO 14900
120  PRINT  CHR$ (5): HGR2 : GOSUB 10000
195  PRINT  CHR$ (5);: REM  ENABLE BIG PRINTING
197  VTAB 23: HTAB 2: PRINT "PRESS ESC TO QUIT.";
199  VTAB 11
200  PRINT : PRINT " HOW MANY SPEAKERS"
202  PRINT " WILL YOU USE ?"
203  PRINT " PRESS 1 OR 2. ";
205  POKE  - 16368,0: GET A$: PRINT A$
206  IF A$ =  CHR$ (27) THEN 15000: REM ESCAPE
207  IF A$ <  > "1" AND A$ <  > "2" THEN  PRINT  CHR$ (7); CHR$ (7): FOR I = 1 TO 333: NEXT I: GOTO 199
220  IF A$ = "2" THEN 500
260  FOR I = 1 TO 7: VTAB 9 + 2 * I: HTAB 1: PRINT "                    ";: NEXT : REM CLEAR BOTTOM OF SCREEN
270  VTAB 11: HTAB 1: PRINT "0  APPLE'S SPEAKER"
275  PRINT "1  EXTERNAL SPEAKER": PRINT "ESC - QUIT FOR NOW."
276  POKE  - 16384,0
277  PRINT : PRINT "PRESS YOUR NUMBER: ";: GET A$: PRINT A$
279  IF  ASC (A$) = 27 THEN 15000: REM   BACK TO THE MENU ON ESCAPE KEY
280 N =  VAL (A$): IF N > 1 OR (N < 1 AND A$ <  > "0") THEN  PRINT  CHR$ (7): GOTO 270
282  IF N = 0 THEN  POKE 778,48: POKE 791,48: REM ADDRESSES TO TURN APPLE'S SPEAKER ON OR OFF
284  IF N = 1 THEN  POKE 778,89: POKE 791,88: REM  ADDRESSES TO TURN AN(0) EXTERNAL SPEAKER ON AND OFF $C059 (LOW BYTE) = 89
290  PRINT  CHR$ (12)
300 A = .000052:B = .00000956:C = .0: REM  FOR FREQUENCY IN LINE 360
304 HT = 8: VTAB 21: PRINT  CHR$ (18): REM ENABLE NORMAL SIZE PRINTING
305  VTAB 21: HTAB HT: PRINT "HIGHER  -  RIGHT ARROW"
306  HTAB HT: PRINT "LOWER   -  LEFT ARROW"
307  HTAB HT: PRINT "ESCAPE  -  QUIT"
308  PRINT  CHR$ (5);: REM  BACK TO BIG PRINT
320  VTAB 7: HTAB 7: PRINT  CHR$ (5);"FREQUENCY": REM LEAVE BIG PRINTING TURNED ON
350  POKE 0,200: POKE 1,0: POKE 2,0: REM  SET UP THE FIRST TONE
355 P =  PEEK (0) + 256 *  PEEK (1)
360 FR =  INT (1 / (A + B * P) + .5):FR$ =  STR$ (FR)
366  VTAB 11: HTAB 9: PRINT "      ": REM CLEAR OUT THE OLD FREQUENCY
370  VTAB 11: HTAB 9 + 2 * ((FR < 10000) + (FR < 1000) + (FR < 100)): PRINT FR$
376  POKE 4,0: REM   FLAG SAYS FREQUENCY ON SCREEN HAS BEEN UPDATED
380  CALL 24586: REM   PLAY TONES WHILE WAITING FOR ARROW KEYS
390 KEY =  PEEK (3)
394  POKE 24587,250
395  IF FR < 200 THEN  POKE 24587,255: REM DELAY BEFORE UPDATING SCREEN FREQUENCY
396  IF FR > 800 THEN  POKE 24587,150
410  IF KEY <  > 27 THEN 355: REM  CODE 27 IS ESCAPE
440  IF KEY = 27 THEN  PRINT  CHR$ (18); CHR$ (12); CHR$ (7);: GOTO 120: REM ESCAPE TO MENU
490  STOP
```

```
500  REM TWO SPEAKERS
505  PRINT  CHR$ (5): HGR2
510  VTAB 1: HTAB 1: PRINT " TWO SPEAKERS"
515  VTAB 5: PRINT "0  SPEAKER ZERO"
517  VTAB 7: PRINT "1  SPEAKER ONE"
519  VTAB 9: PRINT "2  SPEAKER TWO"
521  VTAB 11: PRINT "3  SPEAKER THREE
522  VTAB 13: PRINT "4  APPLE'S INTERNAL"
523  PRINT "          SPEAKER": PRINT "ESC - QUIT"
524  VTAB 21: HTAB 1: PRINT "                    ";
525  VTAB 23: HTAB 1: PRINT "              ";
526  VTAB 21: HTAB 1: PRINT "PRESS A CHOICE: ";
530  POKE  - 16368,0: GET S1$:S1 =  VAL (S1$): PRINT S1$
531  IF S1$ =  CHR$ (27) THEN 120: REM ESCAPE
532  IF S1 > 4 OR (S1$ <  > "0" AND S1 < 1) THEN  VTAB 23: PRINT "  E R R O R"; CHR$ (7); CHR$ (7);: GOTO 524
535 S = S1: GOSUB 990: REM  DRAW A BOX
537  VTAB 23: HTAB 1: PRINT "              ";
538  VTAB 21: HTAB 1: PRINT "                    ";
540  VTAB 21: HTAB 1: PRINT "PRESS A CHOICE: ";
542  POKE  - 16368,0: GET S2$:S2 =  VAL (S2$): PRINT S2$
543  IF S2$ =  CHR$ (27) THEN 120: REM ESCAPE
544  IF S2 > 4 OR (S2$ <  > "0" AND S2 < 1) THEN  VTAB 23: PRINT "  E R R O R"; CHR$ (7); CHR$ (7);: GOTO 537
545  IF S2 = S1 THEN S2 = 99: GOTO 544
550 S = S2: GOSUB 990
560  ON S1 + 1 GOTO 562,563,564,565,566
562 P = 89:Q = 88: GOTO 570: REM  P TURN ON, Q TURN OFF
563 P = 91:Q = 90: GOTO 570
564 P = 93:Q = 92: GOTO 570
565 P = 95:Q = 94: GOTO 570
566 P = 48:Q = 48
570  POKE 861,P: POKE 866,Q
580  ON S2 + 1 GOTO 582,583,584,585,586
582 P = 89:Q = 88: GOTO 590
583 P = 91:Q = 90: GOTO 590
584 P = 93:Q = 92: GOTO 590
585 P = 95:Q = 94: GOTO 590
586 P = 48:Q = 48
590  POKE 901,P: POKE 906,Q
600  PRINT  CHR$ (18): HGR2 : VTAB 1: HTAB 1
605  PRINT "PRESS ZERO BOTH TIMES TO STOP.": PRINT
610  PRINT "INPUT THE FREQUENCY OF THE FIRST SPEAKER(BETWEEN 20 AND 20,000)."
615  PRINT : PRINT "  FREQUENCY: ";: GOSUB 16000:F1$ = II$:F1 =  VAL (F1$)
620  IF F1 < 0 OR F1 > 20000 THEN  PRINT  CHR$ (7): GOTO 600
630  VTAB 12: HTAB 1
633  PRINT "INPUT THE FREQUENCY OF THE SECOND       SPEAKER (BETWEEN 20 AND 20,000)."
636  PRINT : PRINT "  FREQUENCY: ";: GOSUB 16000:F2$ = II$:F2 =  VAL (II$)
637  IF F1 = 0 AND F2 = 0 THEN 500: REM  BOTH FREQUENCIES ZERO
638  IF F2 < 0 OR F2 > 20000 THEN  PRINT  CHR$ (7): VTAB 15: HTAB 1: PRINT "                                        ": GOTO 630
650 A2 = 1.2E - 4:B2 = 5E - 5
655 P1 = 1 / (A2 + B2 * F1): POKE 1,P1 / 256: POKE 0,P1 - 256 *  PEEK (1)
657 P2 = 1 / (A2 + B2 * F2): POKE 3,P2 / 256: POKE 2,P2 - 256 *  PEEK (3)
660  VTAB 21: HTAB 5: PRINT "PRESS ANY KEY TO STOP."
662  HCOLOR= 5: HPLOT 9,158 TO 271,158 TO 271,171 TO 9,171 TO 9,158
670  CALL 816
899  PRINT  CHR$ (12): GOTO 605
990  HCOLOR= 7:Y = 31 + 16 * S
992  HPLOT 0,Y TO 270,Y TO 270,Y + 15 * (1 + (S > 3)) TO 0,Y + 15 * (1 + (S > 3)) TO 0,Y: RETURN
```

```
10000  SPEED= 166
10001  VTAB 1: HTAB 1
10002  PRINT "**********************";
10005  PRINT "*   A U D I O      *";
10010  PRINT "*   OSCILLATOR     *";
10020  PRINT "**********************"

10040  SPEED= 255: FOR I = 1 TO 666: NEXT I: RETURN
10050  REM
14900  PRINT : PRINT  CHR$ (4);"PR#0"
14905  TEXT : HOME : SPEED= 255
14910  PRINT "ERROR - STOPPED AT LINE "; PEEK (218) + 256 *  PEEK (219)
14920  PRINT : PRINT "END": PRINT : END
15000  PRINT : PRINT "PR#0"
15010  TEXT : HOME : PRINT "END"
15015  PRINT : PRINT "LOADING THE MENU."
15020  PRINT  CHR$ (4);"EXEC FRESH START"
15030  END
16000  POKE  - 16368,0: POKE 37496,0
16004 CH =  PEEK (36) + 1:CV =  PEEK (37) + 1: REM  CURSOR
16005 II$ = "": ONERR  GOTO 16073
16008 PP =  PEEK (103):QQ =  PEEK (104)
16009 RR =  PEEK (121) + 256 *  PEEK (122) + 1: POKE 103,RR -  INT (RR / 256) * 256: POKE 104, INT (RR / 256)
16010  GET CC$:LL =  PEEK ( - 16336) -  PEEK ( - 16336) +  PEEK ( - 16336)
16015 LL =  LEN (II$):AA =  ASC (CC$)
16020  IF AA = 3 THEN  POP : POKE 104,QQ: POKE 103,PP: GOTO 14900
16030  IF AA = 21 THEN AA = 32
16040  IF AA = 8 AND LL > 1 THEN  POKE 37496,128: VTAB CV: HTAB CH:II$ =  LEFT$ (II$,LL - 1): PRINT II$;"  ";: VTAB CV: HTAB CH:
 POKE 37496,0: PRINT II$;: GOTO 16010
16050  IF AA = 8 AND LL THEN II$ = "": VTAB CV: HTAB CH: PRINT " ";: VTAB CV: HTAB CH: GOTO 16010
16060  IF (AA = 8) THEN 16010
16070  IF AA <  > 13 THEN 16080: REM     CARRIAGE RETURN
16073  POKE 103,PP: POKE 104,QQ:Z$ = II$:A =  VAL (II$)
16074  ONERR  GOTO 14900
16076  POKE 37496,128: RETURN
16080  IF AA = 24 AND LL THEN  VTAB CV: HTAB CH: FOR LL = 1 TO  LEN (II$): PRINT " ";: NEXT :II$ = "": VTAB CV: HTAB CH: GOTO 16
010
16090  IF AA = 24 THEN 16010
16100  IF AA = 27 THEN 16010
16200 CC$ =  CHR$ (AA):II$ = II$ + CC$
16220  IF AA < 32 THEN 16010
16230  PRINT CC$;: GOTO 16010
63000  PRINT  CHR$ (13); CHR$ (4);"BLOAD GRAPE"
63075  POKE 36223,128
63090  HGR2
63100  CALL 36181: REM  ENABLE GRAPE HI-RES PRINTING
63105  POKE 37496,128: REM       DISABLE CURSOR
63110  RETURN
```

File name: TWO SPEAKER TONES
contains the program below, at $300,
plus the program on the next page at $330.

Description: The program toggles a speaker on and off.
The delay counter at $00, 01 controls the time spent in both the on and off parts of a square wave. The subroutine returns if any key is pressed, or if it makes a number of wave cycles controlled by the counter in $05, 06.

ONE SPEAKER TONES (JSR $300, CALL 768)
$00, 01 = delay counter that controls the frequency
The minimum delay, highest frequency,
is when both locations are zero.
$05, 06 = number of cycles counter. When this counter increases past $FFFF then the subroutine will return.

```
0300-   8D 10 C0    STA   $C010
0303-   E6 00       INC   $00
0305-   A6 00       LDX   $00
0307-   A4 01       LDY   $01
0309-   AD 59 C0    LDA   $C059
030C-   CA          DEX
030D-   D0 FD       BNE   $030C
030F-   88          DEY
0310-   10 FA       BPL   $030C
0312-   A6 00       LDX   $00
0314-   A4 01       LDY   $01
0316-   AD 58 C0    LDA   $C058
0319-   CA          DEX
031A-   D0 FD       BNE   $0319
031C-   88          DEY
031D-   10 FA       BPL   $0319
031F-   E6 05       INC   $05
0321-   D0 04       BNE   $0327
0323-   E6 06       INC   $06
0325-   F0 05       BEQ   $032C
0327-   AD 00 C0    LDA   $C000
032A-   10 D9       BPL   $0305
032C-   C6 00       DEC   $00
032E-   60          RTS
```

AUDIO OSCILLATOR:

Make tones on two speakers at once.

```
0330-  A9 00     LDA  #$00
0332-  85 04     STA  $04
0334-  85 05     STA  $05
0336-  8D 10 C0  STA  $C010
0339-  A5 00     LDA  $00
033B-  8D C0 03  STA  $03C0
033E-  A5 01     LDA  $01
0340-  8D C1 03  STA  $03C1
0343-  A5 02     LDA  $02
0345-  8D C2 03  STA  $03C2
0348-  A5 03     LDA  $03
034A-  8D C3 03  STA  $03C3
034D-  A6 00     LDX  $00
034F-  A4 02     LDY  $02
0351-  CA        DEX
0352-  D0 25     BNE  $0379
0354-  C6 01     DEC  $01
0356-  10 21     BPL  $0379
0358-  A5 04     LDA  $04
035A-  30 05     BMI  $0361
035C-  8D 30 C0  STA  $C030
035F-  10 03     BPL  $0364
0361-  8D 30 C0  STA  $C030
0364-  A9 80     LDA  #$80
0366-  45 04     EOR  $04
0368-  85 04     STA  $04
036A-  AD 00 C0  LDA  $C000
036D-  30 32     BMI  $03A1
036F-  A6 00     LDX  $00
0371-  AD C1 03  LDA  $03C1
0374-  85 01     STA  $01
0376-  EA        NOP
0377-  EA        NOP
0378-  EA        NOP
0379-  88        DEY
037A-  D0 D5     BNE  $0351
037C-  C6 03     DEC  $03
037E-  10 D1     BPL  $0351
0380-  A5 05     LDA  $05
0382-  30 05     BMI  $0389
0384-  8D 59 C0  STA  $C059
0387-  10 03     BPL  $038C
0389-  8D 58 C0  STA  $C058
038C-  A9 80     LDA  #$80
038E-  45 05     EOR  $05
0390-  85 05     STA  $05
0392-  AD 00 C0  LDA  $C000
0395-  30 0A     BMI  $03A1
0397-  A4 02     LDY  $02
0399-  AD C3 03  LDA  $03C3
039C-  85 03     STA  $03
039E-  4C 51 03  JMP  $0351
03A1-  AD 62 03  LDA  $0362
03A4-  8D BA 03  STA  $03BA
03A7-  AD 63 03  LDA  $0363
03AA-  8D BB 03  STA  $03BB
03AD-  AD 8A 03  LDA  $038A
03B0-  8D BD 03  STA  $03BD
03B3-  AD 8B 03  LDA  $038B
03B6-  8D BE 03  STA  $03BE
03B9-  8D 30 C0  STA  $C030
03BC-  8D 58 C0  STA  $C058
03BF-  60        RTS
```

File name: REPEAT TONES
contains the program below which controls increasing and decreasing the frequency for the one speaker tones program at $300.

Description: The program calls $300 to make a tone and wait for an arrow key to be pressed. If a right or left arrow is pressed then the delay counter at $00, 01 is changed. If any other key is pressed then the program returns (back to Applesoft). Also, the program returns to Applesoft if the counter at $05, 06 times out. This allows the user to hold down an arrow key and stay in the tone subroutine, or to let it go and return to Applesoft after a second or two.

$00, 01 = delay counter for subroutine $300
$03 = last key pressed (high byte off) in case of an ESCAPE key
$04 = a flag: if not zero, then the delay counter has been changed since the last time Applesoft updated the screen.
$05, 06 = counter for the number of cycles that will be made before a time-out return to Applesoft.

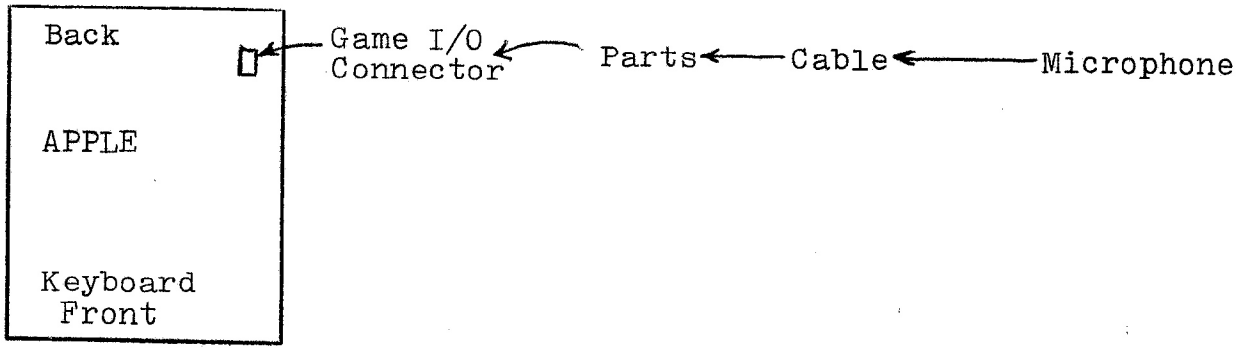
```
6000-   E6 00       INC   $00
6002-   D0 02       BNE   $6006
6004-   E6 01       INC   $01
6006-   A9 FF       LDA   #$FF
6008-   85 04       STA   $04
600A-   A9 FD       LDA   #$FD
600C-   85 06       STA   $06
600E-   20 00 03    JSR   $0300
6011-   AD 00 C0    LDA   $C000
6014-   10 2A       BPL   $6040
6016-   29 7F       AND   #$7F
6018-   85 03       STA   $03
601A-   C9 08       CMP   #$08
601C-   F0 E2       BEQ   $6000
601E-   C9 15       CMP   #$15
6020-   D0 14       BNE   $6036
6022-   C6 00       DEC   $00
6024-   A9 FF       LDA   #$FF
6026-   C5 00       CMP   $00
6028-   D0 DC       BNE   $6006
602A-   C6 01       DEC   $01
602C-   10 D8       BPL   $6006
602E-   A9 00       LDA   #$00
6030-   85 00       STA   $00
6032-   85 01       STA   $01
6034-   10 D0       BPL   $6006
6036-   8D 10 C0    STA   $C010
6039-   60          RTS
603A-   EA          NOP
603B-   EA          NOP
603C-   EA          NOP
603D-   EA          NOP
603E-   EA          NOP
603F-   EA          NOP
6040-   A9 FE       LDA   #$FE
6042-   85 06       STA   $06
6044-   A5 04       LDA   $04
6046-   10 C6       BPL   $600E
6048-   60          RTS
```

MICROPHONE

HOW IT WORKS

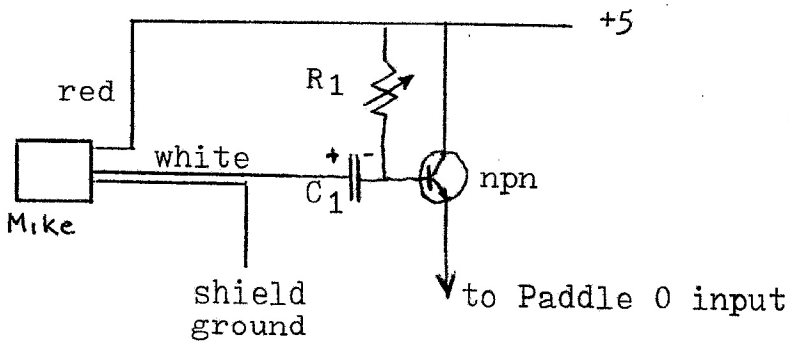
The microphone is an electret, a permanently charged capacitor. Sound vibrations change the distance between the capacitor plates and generate a small AC voltage. The microphone in this kit also amplifies the AC voltage before sending it out on the white wire. This type of microphone has a nearly flat frequency response from 20 to 20,000 hertz.

See the microphone package back for a frequency response chart.



Apple's game paddle input expects current to flow into the PDL0 connection. This current charges up a capacitor inside the computer. A program in the computer counts how long it takes to charge the capacitor and thus measures the current flowing through the game paddle.

Even with its amplifier, the microphone can't supply enough current to drive the game paddle input. We have to add another stage of amplification with one transistor. In the circuit diagram, R1 is a potentiometer that adjusts a small DC current and provides the base-emitter .7 volt bias. The large capacitor C1 passes the AC signal from the microphone. Microphone AC is amplified by the transistor's current gain and then is large enough to affect the game paddle input. Capacitor C1 should be as large as possible to pass more AC into the transistor, but if it is too large it won't fit on the 16 pin header.



ASSEMBLY

The game I/O socket is located at the right rear of the Apple. It has 16 pins. Pin 1 is nearest the keyboard. Check that the plug from this kit fits into the socket.

GAME I/O CONNECTOR

TOP VIEW
(Front Edge of PC Board

+5V	1		16	N.C.
SW0	2		15	AN0
SW1	3		14	AN1
SW2	4		13	AN2
C040 STB	5		12	AN3
PDL0	6		11	PDL3
PDL2	7		10	PDL1
GND	8		9	N.C.

The microphone needs a three wire cable. The cable should be long enough to reach from the Apple to every possible experiment. When it is the right length, strip about one centimeter on both ends of all wires.

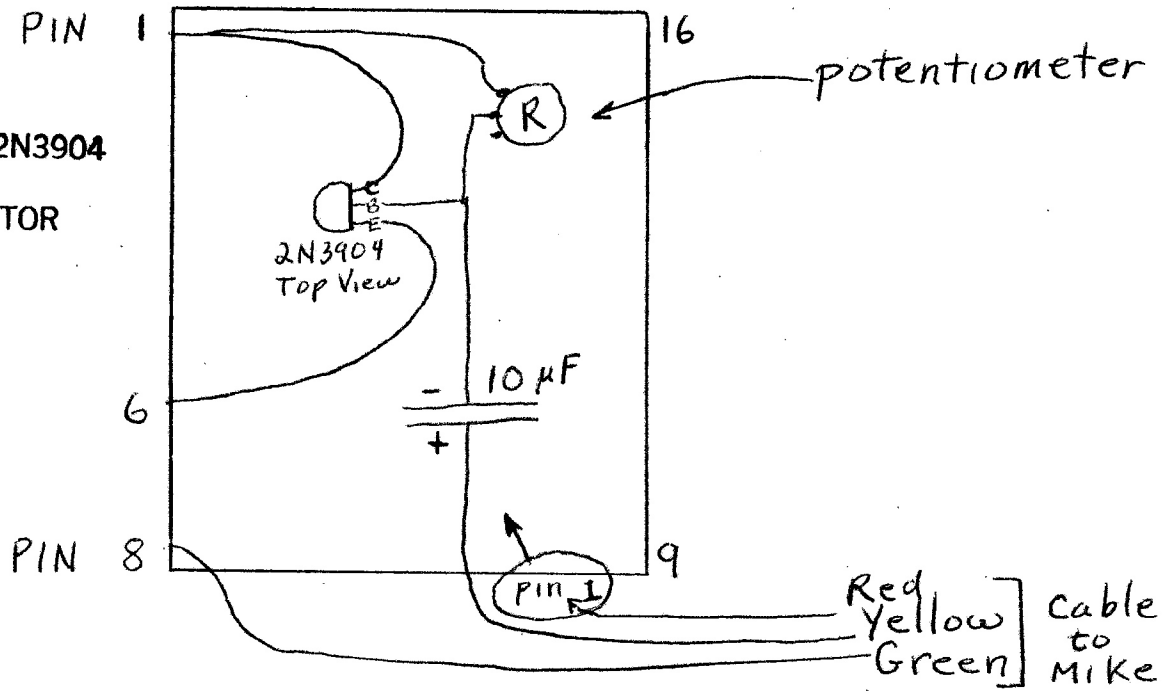
The transistor, capacitor, and potentiometer can all be mounted on the 16 pin header plug included in this kit. This keeps the microphone looking neat, but the cover of the computer must be opened to adjust the potentiometer. Another way is to run three wires from the 16 pin socket to an external circuit board. Mount the transistor, capacitor, and potentiometer on the circuit board. Then run three more wires from the circuit board to the microphone.

276-1603 2N3904

NPN TRANSISTOR

Pin 1 Emitter
Pin 2 Base
Pin 3 Collector

BOTTOM VIEW



TESTING THE MICROPHONE

The potentiometer is a sensitivity adjustment. If it has *high resistance* then very little DC flows through the transistor. The AC from the mike is relativly more important. The game paddle interface will be more sensitive to sounds, but it will take longer to read it because of the small current. On the other hand, if the potentiometer has *low resistance* then more current flows and the interface can be read quickly.

Low potentiometer setting: quick response time,
not sensitive to weak sounds

High potentiometer resistance: slow response,
high sensitivity

Slow response time doesn't hurt when measuring average sound intensity. Fast response is essential when studying the details of a sound wave shape. The absolute fastest the computer can sample the game paddle interface is about 100 microseconds. This can take ten samples during a 1000 hertz wave and get the wave shape. More typically the computer takes 500 microseconds to read the microphone. This just catches the peak and trough of a 1000 hz wave for a frequency measurement. We recommend setting the potentiometer to the middle of its range at first. The sample graphs in the Spectrum Analyzer section below were made with a 10 K potentiometer setting.

```
FOR I = 1 TO 40000: PRINT I, PDL(0): NEXT I
```

One way to set the potentiometer is to type in and run the above line. You will see some game paddle reading between 0 and 255. Turn the pot while watching the numbers on the screen. Stop when the paddle is set between five and ten. Test the sound response by talking into the microphone. The numbers should change while you talk or whistle into the mike.

MEASURING SOUND INTENSITY

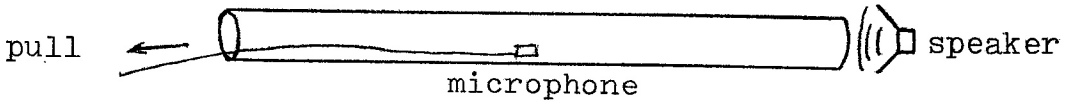
The power in a wave is proportional to the square of its amplitude. The amplitude of our sound wave is the difference between the microphone reading and the DC value when there is no sound. This program very quickly does the following:

1. Read the microphone 256 times.
2. Add up all the data and divide by 256 to get the average.
3. Square the difference between each data point and the average.
4. Add up the squares of the differences and divide by 256.
5. Print the result. Due to round off error the intensity might not be exactly zero when there is no sound.

The Applesoft program is listed on the next page. The machine language program is listed in the spectrum analyzer section. The machine language portion is part of the file FFT.ML and uses two major subroutines from the Fourier transform program. It does the above calculations, steps 1-4, in a fraction of a second. The Applesoft program's function is to call the machine language program and print the sound intensity in the middle of the screen.

We measured the sound intensity while blowing a steady note on a recorder. Any whistle will do. We found that sound does NOT come out the end of a whistle. Most of the noise comes out where air blows past the sharp edge. Also a lot of sound comes out the tone holes.

This program could also study a standing wave pattern in a hollow tube. Place the microphone in the tube and pull it through while reading the sound intensity.



SPEED OF SOUND

Sound moves at the speed 34,000 cm/sec. It takes the Apple an average of .001 seconds to read the game paddle interface. Sound waves move 34 centimeters during one reading. This speed is too fast for measuring by a direct time-of-flight technique.

Speed of a wave equals frequency times wavelength. One can measure the speed of sound from its frequency and wavelength in a resonance tube experiment. Refer to the *Sound Intensity* section above.

DOPPLER EFFECT

This program is a tutorial, not a lab experiment. It shows spherical wave fronts in motion toward and away from an ear. The spacing between the waves changes in each case.

```
1  REM   SOUND INTENSITY AT GAME PADDLE ZERO
2  REM  COPYRIGHT 1983 DR. MARK CROSS

100  HIMEM: 31740
102  GOSUB 63000: REM SET UP THE GRAPE HIGH-RES PRINTER
110  PRINT  CHR$ (13); CHR$ (4);"BLOAD FFT.ML"
115  ONERR  GOTO 14900
120  GOSUB 10000
195  VTAB 13
200  PRINT "THIS PROGRAM READS A MICROPHONE"
202  PRINT "CONNECTED TO GAME PADDLE ZERO AND"
204  PRINT "CONTINUOUSLY PRINTS A NUMBER"
205  PRINT "PROPORTIONAL TO THE POWER (INTENSITY)"
206  PRINT "OF THE SOUND."
208  PRINT : PRINT "   PRESS 'ESC' TO QUIT."
210  PRINT "PRESS ANY OTHER KEY TO START.   ";
220  POKE  - 16368,0: GET A$: PRINT A$
230  IF A$ =  CHR$ (27) THEN  PRINT : PRINT " E S C A P E": PRINT  CHR$ (13); CHR$ (4);"EXEC FRESH START"
240  FOR I = 11 TO 23: VTAB I: HTAB 1: PRINT "                                                ";: NEXT
250  VTAB 23: HTAB 8: PRINT "PRESS 'ESC' TO STOP."
255 C1 = 18674:C2 = 18675:C3 = 18676:C4 = 256:C5 = 256 * 256
260  VTAB 15: HTAB 1: PRINT  CHR$ (5)
270  CALL 18259: REM GET 256 DATA POINTS, ABOUT .1 SECOND
275  REM CALL 18259 ALSO FINDS THE SUM OF THE SQUARES OF THE DIFFERENCE FROM THE AVERAGE SOUND
280 II =  PEEK (C1) + C4 *  PEEK (C2) + C5 *  PEEK (C3)
285 II =  INT ((II + .5) / 100): REM   ROUND OFF
287  VTAB 15: HTAB 5: PRINT "          "
290  VTAB 15: HTAB 15 + (II < 10) + (II < 100) + (II < 1000) + (II < 10000) + (II < 100000)
300  PRINT II;" "
310  IF  PEEK ( - 16384) <  > 155 THEN 270
400  TEXT : PRINT  CHR$ (13); CHR$ (4);"PR#0"
420  HOME : VTAB 5: PRINT "STOPPED  - - -  LOADING THE MENU."
430  PRINT  CHR$ (4);"EXEC FRESH START"
10000  REM  TITLE PAGE
10001  PRINT  CHR$ (5);: VTAB 1: HTAB 1
10002  PRINT "**********************";
10005  PRINT "*    S O U N D     *";
10010  PRINT "*    INTENSITY     *";
10020  PRINT "**********************"
10040  FOR I = 1 TO 999: NEXT I: RETURN
14900  PRINT : PRINT  CHR$ (4);"PR#0"
14905  TEXT : HOME : SPEED= 255
14910  PRINT "ERROR - STOPPED AT LINE "; PEEK (218) + 256 *  PEEK (219)
14920  PRINT : PRINT "END": PRINT : END
63000  PRINT  CHR$ (13); CHR$ (4);"BLOAD GRAPE"
63075  POKE 36223,128
63090  HGR
63100  CALL 36181: REM TURN ON HI-RES PRINTING
63102  IF  PEEK (36045) = 64 THEN  CALL 36611: PRINT  CHR$ (12): REM   SWITCH TO SCREEN 1,  HGR
63105  POKE 37496,128: REM       DISABLE CURSOR
63110  RETURN
```

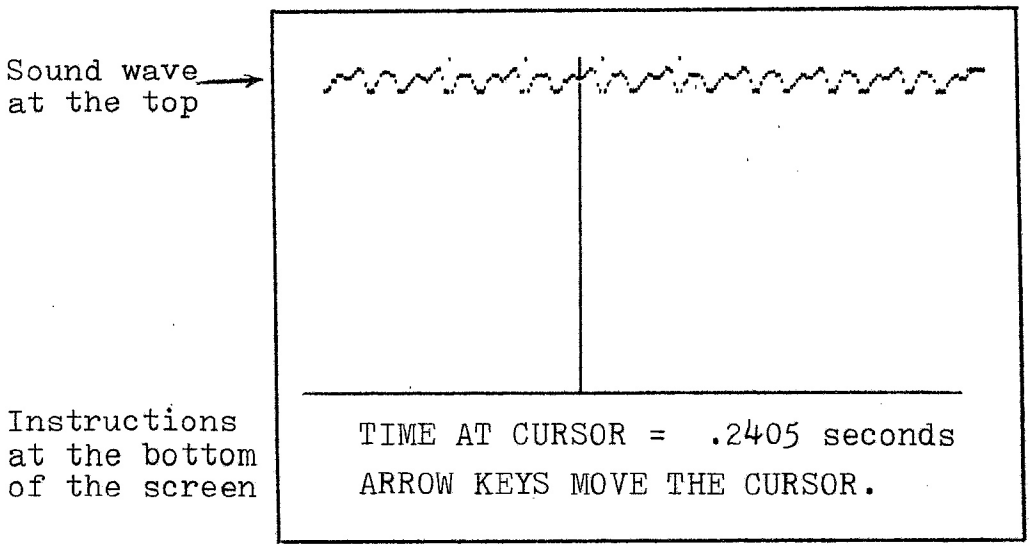
SEE YOUR VOICE

This popular demonstration usually is done by a microphone attached to an oscilloscope. The scope trace vibrates up and down when someone talks.

The imitation Apple oscilloscope isn't as fast as a real scope. Instead we take about ten seconds of sound data all at once and display it slowly so the waves can be studied carefully. The *SEE YOUR VOICE* program waits for a key to be pressed, then fills all available memory with sound data, then displays the wave shape while the user moves a curser across it. The user can examine all of the data in memory. The computer prints the elapsed time as the user moves through the data. The frequency can be measured by moving the cursor to two peaks of the wave, reading the times of both peaks, and calculating one over the time difference.

The computer responds only to low frequencies. A whistle around 1000 hz is near the upper limit of the game paddle interface. The computer can only catch two data points on each 1000 hz wave. Better is to sing a low pitched vowel.

For example we adjusted the potentiometer to have about 7K ohms resistance. We held the microphone very close to a mouth singing 'oooooooo'. The wave, shown below, had a frequency between 200 and 300 hertz.



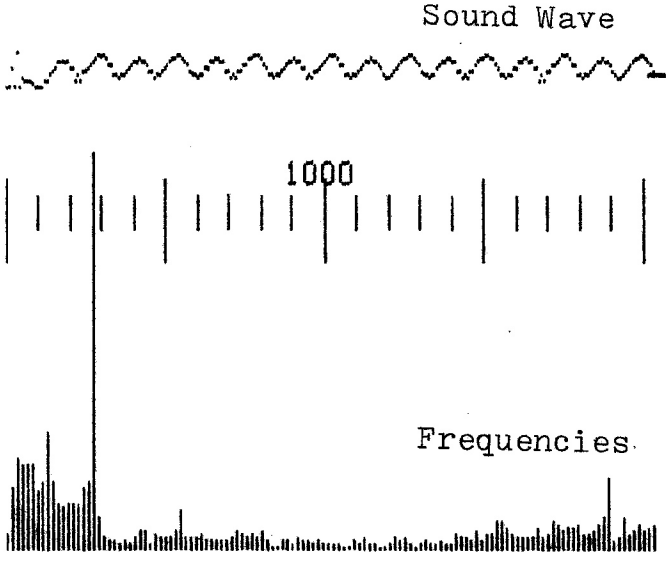
Above is the vowel "ooooo" spoken into a microphone and displayed on Apple's screen. There are about 50 more screens of data extending left and right of this one screen. The user can look through the data and measure the frequency of a wave.

SPECTRUM ANALYZER

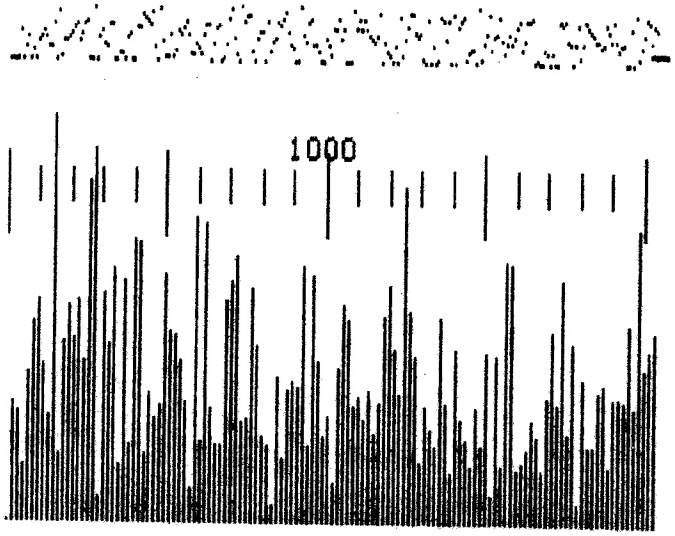
This is the most technically advanced program on the disk. It does a fast Fourier transform of 256 data points while you watch, then displays the frequencies. Two examples are shown below. The vowel 'oooo' was nearly a single frequency around 300 hz, but an 'sssss' was just noise with mixed frequencies.

The fast Fourier transform is described in Creative Computing, July 1980, page 58, and in textbooks. We wrote the transform first in Basic, lines 10-400 of the listing below. It was much too slow so we translated it into machine language. Our machine language fast Fourier transform takes about three seconds to do 256 data points.

When you run the program you will have a choice of using Basic or machine language. The Basic program is not meant for serious use, just for documentation and study. The machine language FFT can work with square, triangle, sine waves, or microphone input. If you chose the microphone then it will take only about a tenth of a second of sound data right after you press the key. Then the computer will just sit there for three seconds working out the frequencies. Finally it will make a Fourier plot like the examples below.



singing "ooooooo"



Hiss - toneless noise

```
2  REM  FAST FOURIER TRANSFORM
3  REM  COPYRIGHT 1983 DR. MARK CROSS
5  GOTO 1000
10 J1 =  INT (J + .1):J3 = 0
11  FOR J4 = 1 TO NU
12 J2 =  INT (J1 / 2)
13 J3 = J3 + J3 + J1 - J2 - J2
14 J1 = J2: NEXT : RETURN
20 CP = 2 * PI / N
25 K = 2 * PI
50  FOR J = 0 TO N
55 C(J) =  COS (CP * J):S(J) =  SIN (CP * J): NEXT
100 N2 = N / 2:N1 = NU - 1:K = 0
110  FOR L = 1 TO NU
115  FOR I = 1 TO N2
120 J = K / (2 ^ N1): GOSUB 10
130 P = J3
135 C = C(P):S = S(P):K1 = K + 1
140 K3 = K1 + N2
145 TR = XR(K3) * C + XI(K3) * S
150 TI = XI(K3) * C - XR(K3) * S
155 XR(K3) = XR(K1) - TR
160 XI(K3) = XI(K1) - TI
165 XR(K1) = XR(K1) + TR
170 XI(K1) = XI(K1) + TI
175 K = K + 1: NEXT I
190 K = K + N2
195  IF K < N THEN 115
200 K = 0:N1 = N1 - 1
300 N2 = N2 / 2
302  NEXT L
305  FOR K = 1 TO N
310 J = K - 1: GOSUB 10:I = J3 + 1
315  IF I < K THEN 350
320 TR = XR(K):TI = XI(K)
325 XR(K) = XR(I):XI(K) = XI(I)
330 XR(I) = TR:XI(I) = TI
350  NEXT K
360 A(0) = XR(1)
365 MAX = 0
370  FOR I = 1 TO N / 2 STEP 2
372 A(I) = XR(I + 1) + XR(N - I + 1)
374 B(I) =  - XI(I + 1) + XI(N - I + 1)
376 P(I) =  ABS (A(I)) +  ABS (B(I))
377  IF P(I) > MAX THEN MAX = P(I)
378  NEXT
400  RETURN
1000  TEXT : HOME : SPEED= 255
1005  LOMEM: 22528: REM $5800
1007  GOSUB 63999: REM SHAPE TABLE
1008  SCALE= 1: ROT= 0: HCOLOR= 7
1010  FOR VT = 3 TO 7: VTAB VT: HTAB 1: PRINT "****************************************";: NEXT
1020  FOR I = 4 TO 6: VTAB I: HTAB 2: PRINT "                                        ": NEXT
1030  VTAB 5: HTAB 9: PRINT "FAST  FOURIER  TRANSFORM"
1050  VTAB 12: HTAB 20: PRINT "COPYRIGHT 1983": HTAB 20: PRINT "DR. MARK CROSS"
1060  DIM XR(256),XI(256),C(256),S(256)
1062  DIM A(256),B(256),P(256)
1063 PI = 3.1415926
1065  PRINT  CHR$ (4);"BLOAD FFT.ML"
1070  VTAB 22: PRINT "    PRESS  ANY  KEY . . . ";
1080  POKE  - 16368,0: GET A$
```

```
1100  HOME : PRINT "THE FOURIER TRANSFORM PICKS OUT THE"
1110  PRINT "FREQUENCIES THAT MAKE UP ANY WAVE."
1120  PRINT : PRINT "THIS PROGRAM LETS YOU PICK A WAVE SHAPE AND FREQUENCY, AND THEN IT FINDS THE"
1130  PRINT "FUNDAMENTAL FREQUENCIES THAT MAKE UP THEWAVE."
1140  PRINT : PRINT "YOU CAN USE A FAST MACHINE LANGUAGE     FOURIER TRANSFORM OR A SLOWER METHOD"
1150  PRINT "WRITTEN IN APPLESOFT BASIC."
1160  PRINT "THE MACHINE LANGUAGE PROGRAM CAN USE    MICROPHONE INPUT, AND THE BASIC PROGRAM"
1170  PRINT "CAN HAVE ANY WAVE SHAPE THAT YOU INSERT INTO THIS PROGRAM.
1200  PRINT : PRINT " 1. MACHINE LANGUAGE
1210  PRINT : PRINT " 2. BASIC"
1220  PRINT : PRINT " 3. QUIT"
1235  VTAB 23: HTAB 1
1240  PRINT "PRESS YOUR NUMBER . . . ";
1250  POKE  - 16368,0: GET A$: PRINT A$;:CH =  VAL (A$)
1260 CH =  VAL (A$): IF CH < 1 OR CH > 3 THEN  PRINT  CHR$ (7);: GOTO 1235
1270  IF CH = 3 THEN  PRINT : PRINT "PROGRAM STOPPED": PRINT : PRINT "LOADING THE MENU.": PRINT  CHR$ (4);"EXEC FRESH START"
1280  IF CH = 2 THEN 2000
1300  REM
USE MACHINE LANGUAGE

1302 NU = 8:N = 2 ^ NU
1305  TEXT
1310  HOME : PRINT "  FAST  FOURIER  TRANSFORM"
1315  PRINT "           MACHINE LANGUAGE"
1320  PRINT : PRINT "WHAT WAVE SHAPE DO YOU WANT?
1330  PRINT : PRINT "1. SQUARE          2. TRIANGLE"
1335  PRINT "3. SINE            4. MICROPHONE INPUT"
1337  PRINT "5. COMPLEX SINE    6. NONE: EXIT"
1340  PRINT : PRINT "PRESS YOUR CHOICE . . . ";
1350  POKE  - 16368,0: GET A$:CH =  VAL (A$): PRINT A$
1352  IF CH < 1 OR CH > 6 THEN  PRINT  CHR$ (7);: GOTO 1310
1353  IF CH > 5.5 THEN 1100
1355  IF CH < 5 THEN 1365
1356  PRINT : PRINT "THE MAIN SINE WAVE HAS FREQUENCY = 1": PRINT "  AND AMPLITUDE = 1."
1357  PRINT : PRINT "YOU CAN ADD A SECOND COMPONENT.": PRINT
1359  PRINT "YOUR FREQUENCY = (1 TO 100): ";: INPUT FRQ
1360  IF FRQ < 1 OR FRQ > 100 THEN  PRINT  CHR$ (7);"ERROR": GOTO 1359
1362  PRINT "YOUR AMPLITUDE = (.001 TO 1): ";: INPUT AMP
1363  IF AMP > 1 THEN  PRINT  CHR$ (7);"ERROR": GOTO 1362
1365  IF CH <  > 4 THEN 1400
1370  PRINT : PRINT "YOU NEED A MICROPHONE CONNECTED TO THE  APPLE'S GAME CONNECTOR.
1375  PRINT : PRINT "PRESS ESCAPE TO RETURN TO THE MENU.
1380  PRINT : PRINT "PRESS ANY OTHER KEY AND THE APPLE WILL  TAKE A FRACTION OF A SECOND OF SOUND."
1385  PRINT "IT WILL ANALYZE THE FREQUENCIES IN THE  SOUND AND MAKE A GRAPH.  ";
1390  POKE  - 16368,0: GET A$
1392  IF  ASC (A$) = 27 THEN 1100
1393  HGR : HOME : VTAB 22: PRINT "    -  W O R K I N G  -"
1394 CC = 8
1395  POKE 17955,76: CALL 17920: REM TRANSFORM ONE SAMPLE
1396  FOR FR = 0 TO 1900 STEP 100:X = FR / CC: HPLOT X,50 TO X,60: NEXT : FOR FR = 0 TO 2000 STEP 500:X = FR / CC: HPLOT X,45 TO X
,70: NEXT
1397 M$ = "500      1000     1500":X9 = 50:Y9 = 40: GOSUB 20000
1399  GOTO 1500
1400 A = 50
1405  PRINT : PRINT "  - MAKING UP THE WAVE SHAPE -"
1410  ON CH GOSUB 8000,8200,8400
1415  IF CH = 5 THEN  GOSUB 8400:FQ = FRQ: GOSUB 8600
1420 X1 = 18944:X2 = X1 + 256:X3 = X2 + 256:X4 = X3 + 256
1428 X0 = X1 - 256
1430  FOR I = 0 TO 255: POKE X0 + I,XR(I): POKE X1 + I,XR(I): POKE X2 + I,0: POKE X3 + I,0: POKE X4 + I,0: NEXT
1440  HGR : HOME : HCOLOR= 7
1450  VTAB 22: PRINT "THE ORIGINAL WAVE IS AT THE TOP OF THE  SCREEN.  - WORKING ON THE FREQUENCIES -"
1455  IF CH = 1 THEN  HPLOT 0,A / 2 TO 0,A TO N / 2,A TO N / 2,0 TO N,0 TO N,A / 2
1460  IF CH > 1 THEN  HPLOT 0,XR(0): FOR I = 1 TO N - 1: HPLOT  TO I,XR(I): NEXT
1470  CALL 20992: CALL 21865: CALL 22053: CALL 18186
```

```
1480  FOR I = 0 TO 50 STEP 10: DRAW 1 AT I * 2,158: NEXT
1485  FOR I = 0 TO 50 STEP 10: VTAB 21: HTAB 1 + I * 2 / 7: PRINT I;: NEXT
1500  HOME : VTAB 22: PRINT "ORIGINAL WAVE ON THE TOP:"
1510  PRINT "   FOURIER TRANSFORM SPECTRUM ON BOTTOM"
1515  IF CH < > 4 THEN 1600
1520 M$ = "SOUND WAVE":X9 = 184:Y9 = 20: GOSUB 20000
1530 M$ = "FREQUENCIES":X9 = 203:Y9 = 115: GOSUB 20000
1600  PRINT "   PRESS ANY KEY TO CONTINUE . . . ";
1620  POKE  - 16368,0: GET A$
1800  GOTO 1305
2000  TEXT : HOME : PRINT "FOURIER TRANSFORM OF A SQUARE WAVE"
2010  PRINT : PRINT "HOW MANY DATA POINTS DO YOU WANT?
2020  PRINT
2030  FOR I = 2 TO 8: PRINT "   ";I;".   ";2 ^ I;"   POINTS"
2035  NEXT
2040  PRINT
2050  PRINT "PRESS A NUMBER FROM 2 THROUGH 8: ";
2055  POKE  - 16368,0: GET A$: PRINT A$
2065 NU =  VAL (A$): IF NU < 2 OR NU > 8 THEN  PRINT  CHR$ (7): GOTO 2050
2070 N = 2 ^ NU
2075  PRINT : PRINT "- MAKING UP THE WAVE -"
2078 A = 30
2080  FOR I = 1 TO N / 2:XR(I) = A:XI(I) = 0: NEXT
2085  FOR I = I TO N:XR(I) = 0:XI(I) = 0: NEXT :XR(0) = 0:XI(0) = 0
2100  HGR
2110  FOR I = 0 TO N: DRAW 1 AT I * 256 / N,XR(I): NEXT
2120  HPLOT 0,XR(0): FOR I = 0 TO N: HPLOT  TO I * 256 / N,XR(I): NEXT
2140  HOME : VTAB 22: PRINT "- WORKING ON THE FREQUENCIES -
2145  PRINT "-  P L E A S E     W A I T  -"
2160  GOSUB 20
2170  FOR I = 0 TO N / 2: HPLOT I * 256 / N,159 TO I * 256 / N,159 - P(I) * 80 / MAX: NEXT
2200  HOME : VTAB 21: PRINT "THE TRANSFORM OF A SQUARE WAVE"
2210  PRINT "HAS ONLY ODD FREQUENCIES."
2220  PRINT "PRESS ANY KEY TO GO ON . . . ";
2230  POKE  - 16368,0: GET A$
2240  TEXT : GOTO 1100
8000 J = N / 2: REM CREATE A SQUARE WAVE
8020  FOR I = 0 TO N:X = A * (I < J)
8030 XR(I) = X::XI(I) = 0: NEXT
8040  RETURN
8200 AA = 20: REM TRIANGLE
8210  FOR I = 0 TO N / 4
8220 X = AA * I * 4 / N:XR(I) = AA + X:XR(I + N / 2) = AA - X
8230 XR(N - I) = AA - X:XR(N / 2 - I) = AA + X
8240  NEXT I
8270  FOR I = 0 TO N:XI(I) = 0: NEXT
8280  RETURN
8400  REM SINE WAVE
8410 FQ = 1:A = 30:AA = .5 * A
8420  FOR I = 0 TO N:XR(I) = A:XI(I) = 0: NEXT
8440  GOTO 8605
8600 AA = AMP * A * .5
8605 TP = 2 * PI / (N / FQ)
8610  FOR I = 0 TO N:XR(I) = XR(I) + AA *  SIN (TP * I): NEXT
8650  RETURN
20000  HCOLOR= 3:L9 =  LEN (M$): REM DRAW STRING M$
20020  FOR L8 = 1 TO L9:L7 =  ASC ( MID$ (M$,L8,1)) - 30
20030  IF L7 = 2 THEN  HCOLOR= 0: DRAW 65 AT X9,Y9: HCOLOR= 3: GOTO 20060
20050  DRAW L7 AT X9,Y9
20060 X9 = X9 + 7: NEXT L8
20070  RETURN
63999 Z9 =  PEEK (121) + 256 *  PEEK (122):Z9 = Z9 + 73: POKE 232,Z9 -  INT (Z9 / 256) * 256: POKE 233,Z9 / 256: RETURN
```

MACHINE LANGUAGE LISTINGS

MEMORY USAGE

$0-7	Temporary storage
800-1FFF	Applesoft program
2000-3FFF	HGR screen
4000-43FF	Fast HGR plotter
4400-44FF	"See Your Voice" subroutines: read the game paddle, fill memory with data, clear the HGR screen, and plot 256 points
4500-45FF	Empty
4600-47C6	FFT controller and short subroutines
4900-49FF	Original sound data f(time)
4A00-4DFF	FFT workspace and Fourier coefficients
4E00-51FF	Cos, sin tables
5200-56FF	FFT subroutine
5700-up	Empty unless the GRAPE Hi-Res printing system is used.

The FFT controller at $4600 is listed below. The main FFT subroutine at $5200 was too long to include in this manual, but its listing is available for $5.00 on special request.

TRANSIENTS

The spectrum analyzer included with this package takes 256 data points, about .1 second, and then does the FFT. It is possible to fill more memory with sound data before doing the FFT. The space from $5600 to $9600 will hold about two to ten seconds of data. Then the data can be Fourier transformed in segments of 256 points. This could show the changing character of a musical note as it decays.

We have not included a program to study transient sounds on the disk with this package. These are usually research projects and ought to use a high speed analog to digital board instead of the game paddle interface. If a user wants to write a program for transient spectra, he should follow these steps:

1. Read the audio data into $5700-95FF.
2. Move 256 points into $4A00-4AFF.
3. Zero locations $4B00-4DFF.
4. JSR $5200 to do the FFT.
5. JSR $5569 to unshuffle the coefficients.
6. Find the standard Fourier coefficients in these locations:
   - $4A00-4AFF cosine terms, low bytes, starting with DC at $4A00; first frequency at $4A01; second at 4A02
   - #4B00-4BFF cosine terms, high bytes
   - #4C00-4CFF sine terms, low bytes
   - #4D00-4DFF sine terms, high bytes

```
1  ;
2  ;FFT CONTROL PROGRAM
3  ;
4  M0     EPZ $00
5  M1     EPZ $01
6  M2     EPZ $02
7  M3     EPZ $03
8  FLAG   EQU $48E6
9  TMP0   EQU $48E7
10 TMP1   EQU $48E8
11 TMP2   EQU $48E9
12 TMP3   EQU $48EA
13 I      EQU $48EB
14 TI0    EQU $48EC
15 TI1    EQU $48ED
16 TI2    EQU $48EE
17 TI3    EQU $48EF
18 N1     EQU $48F0
19 N2     EQU $48F1
20 K      EQU $48F2
21 K1     EQU $48F3
22 K3     EQU $48F4
23 J      EQU $48F5
24 J3     EQU $48F6
25 L      EQU $48F7
26 CL     EQU $48F8
27 CH     EQU $48F9
28 SL     EQU $48FA
29 SH     EQU $48FB
30 TR0    EQU $48FC
31 TR1    EQU $48FD
32 TR2    EQU $48FE
33 TR3    EQU $48FF
34 PDLDAT EQU $4900
35 XRL    EQU $4A00
36 XRH    EQU $4B00
37 XIL    EQU $4C00
38 XIH    EQU $4D00
39 STABL  EQU $4E00
40 STABH  EQU $4F00
41 CTABL  EQU $5000
42 CTABH  EQU $5100
43 MULT   EQU $5489
44 COEF   EQU $5569
45 POWER  EQU $5625
46 ;
```

```
4600           47        ORG $4600
4600           48        OBJ $800
4600 8D10C0    49        STA $C010
4603 204246    50 CTRL4  JSR READ          ;READ 256 PADDLES, STORE AT $4900, AND COPY TO $4A00 WORKSPACE
4606 200052    51        JSR $5200         ;DO THE FFT OF DATA AT $4A00
4609           52 ;                         AND STORE COEFFICIENTS AT $4A00-4DFF
4609 206955    53        JSR COEF          ;UNSHUFFLE TO GET STANDARD COEFFS
460C A900      54        LDA #$00
460E 8D004A    55        STA $4A00         ;KILL THE DC COEFFICIENT
4611 8D004B    56        STA $4B00
4614 8D004C    57        STA $4C00
4617 8D004D    58        STA $4D00
461A 202556    59        JSR POWER         ;GET ABS(COS) PLUS ABS(SIN) COEFFICIENT INTO $4C00
461D 203547    60        JSR CLEAR         ;CLEAR TOP OF HGR SCREEN
4620 20F646    61        JSR TIMPLT        ;PLOT F(T)  256 POINTS FROM $4900
4623 200A47    62        JSR FFTPLT        ;PLOT ABS (COS) + ABS (SIN) COEFFICIENTS FROM $4C00
4626           63 ; CHANGE THE ABOVE LINE TO JMP FFTPLT AND THE SUBROUTINE WILL REPEAT
4626           64 ; AUTOMATICALLY UNTIL ESC KEY TO ESCAPE, OR 'S' KEY IS PRESSED FOR
4626           65 ; SINGLE STEP OF JUST ONE AUDIO SAMPLE AT A TIME.
4626 AD00C0    66        LDA $C000
4629 C99B      67        CMP #$9B
462B D004      68        BNE CTRL1
462D 8D10C0    69        STA $C010
4630 60        70        RTS
4631 C9D3      71 CTRL1  CMP #$D3          ; LETTER S
4633 D0CE      72        BNE CTRL4
4635 8D10C0    73        STA $C010
4638 AD00C0    74 CTRL3  LDA $C000
463B 10FB      75        BPL CTRL3
463D C99B      76        CMP #$9B
463F D0C2      77        BNE CTRL4
4641 60        78        RTS
```

```
4642          79  ;
4642          80  ; READ 256 PADDLE POINTS
4642          81  ;
4642 A949     82  READ   LDA #$49          ;LOCATION TO STORE THE DATA
4644 8501     83         STA $01
4646 A900     84         LDA #$00
4648 8500     85         STA $00
464A A8       86         TAY
464B          87  ; BELOW IS AN UNWRAPPED LOOP TO READ THE GAME PADDLE UP TO A MAXIMUM
464B          88  ; COUNT OF 17 (DEC).
464B AD70C0   89  READ4  LDA $C070
464E A200     90         LDX #$00
4650 AD64C0   91         LDA $C064
4653 1069     92         BPL READ1
4655 E8       93         INX
4656 AD64C0   94         LDA $C064
4659 1063     95         BPL READ1
465B E8       96         INX
465C AD64C0   97         LDA $C064
465F 105D     98         BPL READ1
4661 E8       99         INX
4662 AD64C0  100         LDA $C064
4665 1057    101         BPL READ1
4667 E8      102         INX
4668 AD64C0  103         LDA $C064
466B 1051    104         BPL READ1
466D E8      105         INX
466E AD64C0  106         LDA $C064
4671 104B    107         BPL READ1
4673 E8      108         INX
4674 AD64C0  109         LDA $C064
4677 1045    110         BPL READ1
4679 E8      111         INX
467A AD64C0  112         LDA $C064
467D 103F    113         BPL READ1
467F E8      114         INX
4680 AD64C0  115         LDA $C064
4683 1039    116         BPL READ1
4685 E8      117         INX
4686 AD64C0  118         LDA $C064
4689 1033    119         BPL READ1
468B E8      120         INX
468C AD64C0  121         LDA $C064
468F 102D    122         BPL READ1
4691 E8      123         INX
4692 AD64C0  124         LDA $C064
4695 1027    125         BPL READ1
4697 E8      126         INX
4698 AD64C0  127         LDA $C064
469B 1021    128         BPL READ1
469D E8      129         INX
469E AD64C0  130         LDA $C064
46A1 101B    131         BPL READ1
46A3 E8      132         INX
46A4 AD64C0  133         LDA $C064
46A7 1015    134         BPL READ1
46A9 E8      135         INX
46AA AD64C0  136         LDA $C064
46AD 100F    137         BPL READ1
46AF E8      138         INX
46B0 AD64C0  139         LDA $C064
46B3 1009    140         BPL READ1
46B5 E8      141         INX
46B6 AD64C0  142         LDA $C064
46B9 1003    143         BPL READ1
46BB 8A      144         TXA
46BC 1007    145         BPL READ2
46BE 8A      146  READ1   TXA
46BF         147  ; MAKE EACH READ LAST THE SAME LENGTH OF TIME
46BF E8      148  READ3   INX
46C0 E014    149         CPX #$14
46C2 EA      150         NOP
46C3 30FA    151         BMI READ3
46C5 9100    152  READ2   STA ($00),Y
46C7 C8      153         INY
46C8 F003    154         BEQ RDON
46CA 4C4B46  155         JMP READ4
```

```
46CD          156 ;
46CD          157 ;MOVE DATA TO FFT WORK AREA
46CD          158 ;ORIGINAL DATA AT $4900, COPY OF DATA AT $4A00
46CD          159 ;ZEROS AT $4B00-4DFF
46CD          160 ;
46CD A94A     161 RDON   LDA #$4A
46CF 8503     162        STA $03
46D1 A000     163        LDY #$00
46D3 8402     164        STY $02
46D5 B100     165 READ5  LDA ($00),Y
46D7 9102     166        STA ($02),Y
46D9 C8       167        INY
46DA D0F9     168        BNE READ5
46DC A94B     169        LDA #$4B
46DE 8501     170        STA $01
46E0 A94C     171        LDA #$4C
46E2 8503     172        STA $03
46E4 A94D     173        LDA #$4D
46E6 8505     174        STA $05
46E8 8404     175        STY $04
46EA A900     176        LDA #$00
46EC 9100     177 READ6  STA ($00),Y
46EE 9102     178        STA ($02),Y
46F0 9104     179        STA ($04),Y
46F2 C8       180        INY
46F3 D0F7     181        BNE READ6
46F5 60       182        RTS
46F6          183 ;
46F6          184 ; PLOT THE TIME DATA
46F6          185 ;
46F6 A949     186 TIMPLT LDA #$49
46F8 8501     187        STA $01
46FA A200     188        LDX #$00
46FC 8600     189        STX $00
46FE 8A       190 TIM1   TXA
46FF A8       191        TAY
4700 B100     192        LDA ($00),Y
4702 A8       193        TAY
4703 20C040   194        JSR $40C0
4706 E8       195        INX
4707 D0F5     196        BNE TIM1
4709 60       197        RTS
```

```
470A          198 ;
470A          199 ;FFT PLT
470A          200 ;PLOTS THE SPECTRAL DATA
470A          201 ;AS A BAR CHART
470A          202 ;
470A A000     203 FFTPLT LDY #$00
470C 8400     204        STY $00
470E 8402     205        STY $02
4710 A94C     206        LDA #$4C
4712 8501     207        STA $01
4714 A502     208 FPLT2  LDA $02
4716 A8       209        TAY
4717 0A       210        ASL
4718 AA       211        TAX
4719 B100     212        LDA ($00),Y
471B 4A       213        LSR
471C 8503     214        STA $03            ;DATA DIVIDED BY TWO
471E A99F     215        LDA #$9F
4720 38       216        SEC
4721 E503     217        SBC $03
4723 A8       218        TAY
4724 20C040   219 FPLT1  JSR $40C0
4727 C8       220        INY
4728 C0A0     221        CPY #$A0
472A D0F8     222        BNE FPLT1
472C E602     223        INC $02
472E A502     224        LDA $02
4730 C981     225        CMP #$81
4732 D0E0     226        BNE FPLT2
4734 60       227        RTS
```

```
4735          228 ;
4735          229 ;CLEAR THE HGR SCREEN
4735          230 ;
4735 A09F     231 CLEAR  LDY #$9F
4737 A227     232 CLR2   LDX #$27
4739 B90040   233        LDA $4000,Y
473C 8D4947   234        STA CLR1+2
473F B94041   235        LDA $4100,Y
4742 8D4847   236        STA CLR1+1
4745 A900     237        LDA #$00
4747 9DFFFF   238 CLR1   STA $FFFF,X
474A CA       239        DEX
474B 10FA     240        BPL CLR1
474D 88       241        DEY
474E C0FF     242        CPY #$FF
4750 D0E5     243        BNE CLR2
4752 60       244        RTS
```

```
4753          245 ;
4753          246 ; SOUND INTENSITY
4753          247 ; AVG OF SQUARES OF SOUND MINUS AVG SOUND
4753          248 ;
4753 204246   249 INT    JSR READ
4756 A900     250        LDA #$00
4758 8500     251        STA M0
475A 8501     252        STA M1
475C 8502     253        STA M2
475E 8DF248   254        STA K
4761 8DF348   255        STA K1
4764 8DF448   256        STA K3
4767 A8       257        TAY
4768 A949     258        LDA #$49
476A 8503     259        STA $03
476C 18       260 INTA   CLC
476D B90049   261        LDA PDLDAT,Y
4770 6500     262        ADC M0
4772 8500     263        STA M0
4774 A900     264        LDA #$00
4776 6501     265        ADC M1
4778 8501     266        STA M1
477A C8       267        INY
477B D0EF     268        BNE INTA
477D A651     269        LDX M1              ;AVERAGE OF ALL 256 DATA POINTS
477F 8EF048   270        STX N1
4782 A000     271        LDY #$00
4784 8CEB48   272 INTB   STY I
4787 38       273        SEC
4788 ADF048   274        LDA N1
478B F90049   275        SBC PDLDAT,Y
478E 1005     276        BPL INTC
4790 49FF     277        EOR #$FF
4792 18       278        CLC
4793 6901     279        ADC #$01
4795 8504     280 INTC   STA $04
4797 8506     281        STA $06
4799 A900     282        LDA #$00
479B 8505     283        STA $05
479D 8507     284        STA $07
479F 208954   285        JSR MULT
47A2 18       286        CLC
47A3 A500     287        LDA M0              ;LOWEST BIT OF SQUARE
47A5 6DF248   288        ADC K
47A8 8DF248   289        STA K
47AB A501     290        LDA M1              ;SECOND LOWEST BIT OF SQUARE
47AD 6DF348   291        ADC K1
47B0 8DF348   292        STA K1
47B3 A502     293        LDA M2
47B5 6DF448   294        ADC K3
47B8 8DF448   295        STA K3
47BB ACEB48   296        LDY I
47BE C8       297        INY
47BF D0C3     298        BNE INTB
47C1 ADF448   299        LDA K3
47C4 8500     300        STA M0              ;SUM OF SQUARES/256 PASSED IN LOCATION ZERO
47C6 60       301        RTS
```

```
** ABSOLUTE VARABLES/LABELS

FLAG   48E6  TMP0   48E7
TMP1   48E8  TMP2   48E9  TMP3   48EA  I      48EB  TI0    48EC  TI1    48ED
TI2    48EE  TI3    48EF  N1     48F0  N2     48F1  K      48F2  K1     48F3
K3     48F4  J      48F5  J3     48F6  L      48F7  CL     48F8  CH     48F9
SL     48FA  SH     48FB  TR0    48FC  TR1    48FD  TR2    48FE  TR3    48FF
PDLDAT 4900  XRL    4A00  XRH    4B00  XIL    4C00  XIH    4D00  STABL  4E00
STABH  4F00  CTABL  5000  CTABH  5100  MULT   5489  COEF   5569  POWER  5625
CTRL4  4603  CTRL1  4631  CTRL3  4638  READ   4642  READ4  464B  READ1  46BE
READ3  46BF  READ2  46C5  RDON   46CD  READ5  46D5  READ6  46EC  TIMPLT 46F6
TIM1   46FE  FFTPLT 470A  FPLT2  4714  FPLT1  4724  CLEAR  4735  CLR2   4737
CLR1   4747  INT    4753  INTA   476C  INTB   4784  INTC   4795
```

Apple Audio Processing

Mark A Cross
Physics Department
Grambling State University
Grambling LA 71245

Tired of poking single tones into your speaker? The Apple is capable of talking or playing several notes simultaneously. It can be done in one evening from very simple homebrew interfaces.

There are at least three ways to get speech out of an Apple. The APPLE-TALKER program by Bob Bishop accepts voice from the cassette input, processes and stores the data, and then pokes it to the internal speaker. A second way is to use a voice synthesizer built on a plug-in card, such as the one made by Mountain Hardware. The third method is described in this article.

The references give the theory behind the methods of analog-to-digital (A/D), input, data storage, digital-to-analog (D/A), and output. They emphasize high sampling rates. Yes, it would be best to sample the input at 100 kHz and store it with 12-bit accuracy to create a high-fidelity computer. This is needed for music, but we are accustomed to sloppy speech. We can sample speech at 2000 Hz, store the data, and send it out to a 4-bit digital-to-analog converter. This reproduces speech which sounds very similar to that reproduced by a tape recorder!

Audio Input

The Apple has four game paddle inputs. These generate a count from 0 to 255 in response to a resistance from 0 to about 130 k ohms. The internal circuit shown in figure 1 has a 553 timer which discharges the 0.022 μF capacitor in response to a LDA $C070 instruction. Then a software counter runs while the capacitor is charged by the +5 V supply at a rate set by the paddle 0 resistance. When the capacitor reaches a trigger voltage, the 553 changes state and the counter stops. The program sequence used to create the counter is as follows:

```
label  mnemonic  operand  comment
       LDA       $C070    Discharge capacitor.
       LDY       #00      Initialize count.
       NOP
       NOP
READ   LDA       $C064    Check status of 553
                           timer.
       BPL       DONE
       INY
       BNE       READ
       DEY
DONE
```

The execution time of this subroutine is a function of register Y. It takes the time $t = 16 + (10 \times Y)$ μs to execute. Suppose that $Y = 7$. Then the rate of cycling through the counter is $f = 1/t$ or approximately 11,600 Hz, minus overhead for storing the data. Speech at 100 to 1000 Hz is well within this sampling rate. Low fidelity music is also possible.

Figure 2 shows how to build a very simple amplifier that will convert an audio input into a variable resistance. The microphone should be a moving coil type. About 10 mV will be generated by the inexpensive microphones that used to be included with cassette recorders, or you can simply talk into a loudspeaker. The input capacitor should be 0.1 μF or more, nonpolarized. If the input capacitor is too small then it will block most of the input. The transistor is any NPN type out of a spare parts box (such as a 2N2222).

I used a 2 M ohm potentiometer for the base resistor. It will be adjusted later to allow for variations between transistors. You might want to include a 100 k ohm fixed resistance in series with the variable 2 M ohm resistance to prevent adjusting the base resistance to zero and destroying the transistor.

The base resistance allows a small current to flow that is amplified by the transistor to make a larger collector current. Both currents flow through the emitter to charge the internal 0.022 μF capacitor. Thus, the steady state of this imitation game paddle can be set by adjusting the base resistance. When you apply a small AC voltage from the microphone, the base current changes. This in turn changes the paddle's effective resistance.

The input circuit can be built on a 16-pin socket as suggested on page 118 of the *Apple II Reference Manual* (the red book). It is difficult to adjust the resistance R and capacitance if you do this. You can also connect two wires from pins 1 and 6 of the game paddle connector and build the amplifier on a breadboard.

Check out the amplifier in BASIC while running line 10 of listing 1 below. Adjust R to get a steady 7 or 8 paddle reading, for the fastest sampling. (Half of fifteen, for the 4-bit output to be used, equals the DC level before the you start talking.) A range of at least 4 units (8 is most desirable) change in PDL (0) caused by your speech is needed. Yell into the mike and hit control-C. You will get more gain by adjusting the base resistance to be larger, or by increasing the input capacitor value.

Text continued on page 216

Listing 1: *Integer BASIC routines for testing the audio-input interface and manipulating the stored data. The routine starting on line 1000 produces a record of different numbers in the raw data. Note the minimum and maximum for later use. Lines 2000 thru 2080 scale the waveforms into the range 0 to 15. First, the minimum is subtracted from every data point to shift it down to 0. Then the wave is either clipped or compressed to bring the maximum down to 15. Lines 3000 thru 3050 send the audio data to the output trying all possible delays. The routine starting at line 4000 compresses the data by discarding every other data point. Lines 5000 thru 5040 show how to call the input subroutine.*

```
>LIST
   10 PRINT PDL (0): GOTO 10: REM  TEST THE INPUT AMPLIFIER
  900 REM
 1000 DIM N(80): REM  STUDY THE AUDIO DATA
 1010 FOR I=0 TO 80:N(I)=0: NEXT I
 1020 FOR I=2816 TO 12287: REM  AUDIO DATA AREA
 1030 X= PEEK (I):N(X)=N(X)+1: NEXT I
 1040 PRINT "I     N(I)      N(20+I)  N(40+I) N(60+I)"
 1050 FOR I=0 TO 19
 1060 PRINT I,N(I),N(20+I),N(40+I),N(60+I)
 1070 NEXT I: END
 1900 REM
 1990 REM  INPUT THE SPEECH DATA
 2000 FOR I=0 TO 80:N(I)=0: NEXT I
 2010 INPUT "MINIMUM DATA ",MIN
 2020 INPUT "MAX DATA ",MAX
 2030 FOR I=2816 TO 12287
 2040 X= PEEK (I)-MIN
 2050 X=X*15/(MAX-MIN): REM  COMPRESSING
 2060 IF X>15 THEN X=15: REM  CLIPPING
 2070 N(X)=N(X)+1: POKE I,X
 2080 NEXT I: GOTO 1040
 3000 INPUT "TURN ON AMPLIFIER AND PRESS RETURN.",A$
 3010 FOR DELAY=0 TO 255
 3020 PRINT DELAY
 3030 POKE 2561,0: POKE 2562,12
 3040 POKE 2612,DELAY: CALL 2560
 3050 NEXT DELAY: END
 3900 REM
 3990 REM  COMPRESS THE DATA BY DISCARDING HALF OF IT
 4000 X=(12287-2816)/2: REM  HALF OF DATA AREA
 4010 FOR I=1 TO X
 4020 POKE 2816+I, PEEK (2816+2*I)
 4030 NEXT I: END
 4900 REM
 5000 REM  CALL INPUT SUBROUTINE
 5010 INPUT "HIT RETURN WHEN READY TO TALK.",A$
 5020 POKE 2325,0: POKE 2326,11
 5030 POKE 2346,0: POKE 2339,48
 5040 POKE 2321,13: CALL 2304: END
```

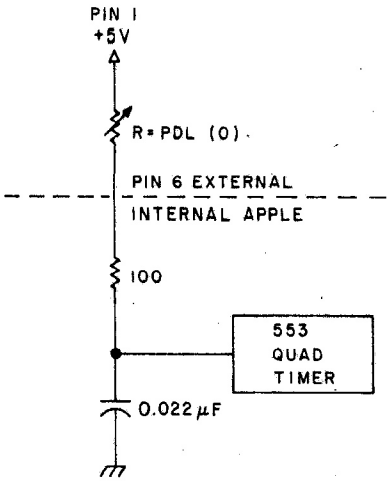


Figure 1: *A representation of the paddle-input system used by the Apple II computer.*

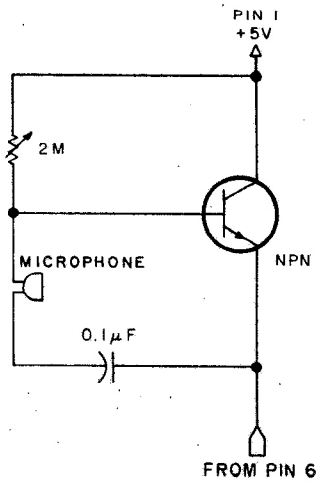


Figure 2: *A microphone and simple amplifier can be added to the Apple paddle connector and used to input audio information. The program in listing 2 is used with this circuit.*

Text continued from page 212:

BASIC cannot sample the game paddle fast enough to follow sounds. The program in listing 2 will do that. Hexadecimal locations 0900 thru 0912 loop indefinitely waiting for the user's initial voice input. When the paddle count reaches (THRESH + 1) (THRESH is threshold to start recording data), the rest of the program begins sensing and storing data. The user can insert a delay loop at hexadecimal 093E to wait between data points and get more (but lower quality) speech into memory.

A standard 16 K byte memory holds one or two words of good quality speech. You can adjust the base resistance in the amplifier to make a large steady PDL (0) value of 50 or more and thus sample the input more slowly. "Row, row, row your boat gently down the stream" will fit in, but the rest of the song might be too noisy if compressed into 16 K bytes.

Listing 2: *6502 assembly-language program to drive the audio-input interface. This reads the voice data from a microphone connected to a game paddle. The data is stored in locations START thru END. When ENDHI equals 48 (decimal), then LOMEM:12289 will put all BASIC work above the audio data area. There are several adjustable parameters: THRESH: threshold to start recording data. Should be 2 or 3 units above the steady state, no-speech PDL(0). STARTLO, STARTHI: start of the data storage area. ENDLO, ENDHI: = LOMEM − 1: end of audio area.*

```
0900- AD 70 C0    LDA   $C070
0903- A0 00       LDY   #$00
0905- EA          NOP
0906- EA          NOP
0907- AD 64 C0    LDA   $C064
090A- 10 04       BPL   $0910
090C- C8          INY
090D- D0 F8       BNE   $0907
090F- 88          DEY
0910- C0 10       CPY   #$10
0912- 30 EC       BMI   $0900
0914- 8C 00 0B    STY   $0B00
0917- EE 15 09    INC   $0915
091A- D0 03       BNE   $091F
091C- EE 16 09    INC   $0916
091F- AD 16 09    LDA   $0916
0922- C9 30       CMP   #$30
0924- 30 08       BMI   $092E
0926- AD 15 09    LDA   $0915
0929- C9 00       CMP   #$00
092B- D0 01       BNE   $092E
092D- 60          RTS
092E- AD 70 C0    LDA   $C070
0931- A0 00       LDY   #$00
0933- EA          NOP
0934- EA          NOP
0935- AD 64 C0    LDA   $C064
0938- 10 04       BPL   $093E
093A- C8          INY
093B- D0 F8       BNE   $0935
093D- 88          DEY
093E- 4C 14 09    JMP   $0914
0941- FF          ???
0942- 00          BRK
0943- 00          BRK
0944- FF          ???
0945- FF          ???
0946- 00          BRK
0947- 00          BRK
0948- FF          ???
```

Processing

After the waveform data is stored in memory it can be easily improved, condensed, or distorted. Try the short programs in Tom O'Haver's article (see references). Keep in mind that the 4-bit output requires all data to be in the range 0 to 15.

The routines in listing 1 can be used to scale, compress, and output the data.

Output

The game connector has four annunciator outputs. These are compatible with the 4-bit digital-to-analog converter shown in figure 3. Build it on the socket that the input amplifier is connected to.

The idea of using a resistor network for digital-to-analog conversion is discussed by Hal Chamberlin (see references). The minimum resistance here is 5 k ohms so that the maximum current drawn from the annunciator outputs will be 1 mA. High-precision resistors are not necessary. The digital-to-analog conversion truncates the fifth bit, which introduces a 3% error. Five-percent tolerance resistors will do.

The capacitor in figure 3 filters out high-frequency noise. The noise comes from truncation to 4 bits, from delays between taking samples of the audio input, and from not changing all 4 bits of the digital output simultaneously. A larger capacitor on the output will filter out more noise, but it will also attenuate the signal, thus, you will have to turn up the amplifier's gain. A better low-pass filter would help.

The output software is shown in listing 3. It fetches a byte of waveform data, sends it to the digital-to-analog converter, increments and tests the memory pointer, waits for a delay, and then fetches another byte of data.

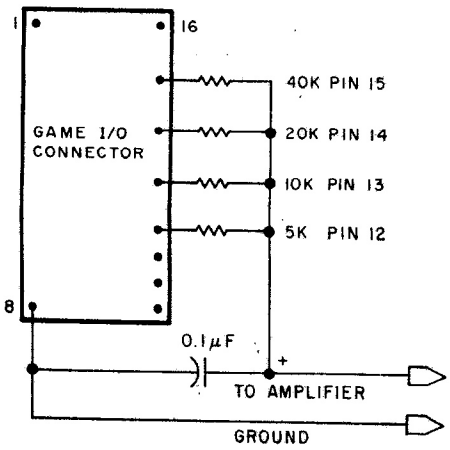


Figure 3: *This simple 4-bit digital-to-analog (D/A) converter, along with listing 3, is used to output information created by the circuit shown in figure 2 and the program shown in listing 2.*

Listing 3: *6502 assembly-language program that sends audio data to the 4-bit digital-to-analog (D/A) converter.*

```
*A00LL
0A00- AD 00 0B    LDA   $0B00
0A03- 6A          ROR
0A04- B0 05       BCS   $0A0B
0A06- 8D 58 C0    STA   $C058
0A09- 90 04       BCC   $0A0F
0A0B- 8D 59 C0    STA   $C059
0A0E- EA          NOP
0A0F- 6A          ROR
0A10- B0 05       BCS   $0A17
0A12- 8D 5A C0    STA   $C05A
0A15- 90 04       BCC   $0A1B
0A17- 8D 5B C0    STA   $C05B
0A1A- EA          NOP
0A1B- 6A          ROR
0A1C- B0 F5       BCS   $0A13
0A1E- 8D 5C C0    STA   $C05C
0A21- 90 04       BCC   $0A27
0A23- 8D 5D C0    STA   $C05D
0A26- EA          NOP
0A27- 6A          ROR
0A28- B0 05       BCS   $0A2F
0A2A- 8D 5E C0    STA   $C05E
0A2D- 90 04       BCC   $0A33
0A2F- 8D 5F C0    STA   $C05F
0A32- EA          NOP
0A33- A2 1E       LDX   #$1E
0A35- CA          DEX
0A36- D0 FD       BNE   $0A35
0A38- EE 01 0A    INC   $0A01
0A3B- D0 03       BNE   $0A40
0A3D- EE 02 0A    INC   $0A02
0A40- AD 02 0A    LDA   $0A02
0A43- C9 30       CMP   #$30
0A45- D0 05       BNE   $0A4C
0A47- AD 01 0A    LDA   $0A01
0A4A- C9 00       CMP   #$00
0A4C- D0 B2       BNE   $0A00
0A4E- 60          RTS
0A4F- 00          BRK
0A50- FF          ???
```

Conclusion

The speech quality produced by this method is relatively good. Most music doesn't turn out very well when the high frequencies are filtered out. I tried "The Star Spangled Banner" from the article by Hal Chamberlin. The music was tolerable but my simple capacitor filter let through too much high-frequency noise (reference 3).

An 8-bit digital-to-analog converter can be built. I did so, but found that it resulted in no significant audible difference for speech. Such an option might be advantageous only if you are interested in high-fidelity music reproduction.

The main problem is the available memory which limits the amount of audio information that can be stored. Slower sampling can store more data, but this introduces too much noise when the sampling rate falls below 1000 to 2000 Hz. You can double up and store 2 units of data in 1 byte of memory. I have been able to get phonemes (eg: single letter sounds) compressed to 256 bytes of memory on the average.

The input routine in listing 2 could be improved. The routine now spends less time sampling low-amplitude inputs and more time sampling high-amplitude inputs. There should be another counter that waits during a variable interval depending on the input amplitude, which is indicated by register Y.

You can change the amplitude of the waveforms. Either divide all the data by 2 in BASIC, or insert an extra rotate right (ROR) instruction in the output routine just before the data gets to the digital-to-analog conversion section. The speech is still intelligible when it is cut down to 2 or 3 bits of data! A better output routine would have a parameter to choose full, ¾, ½, or ¼ amplitude. (Of course this won't work when the audio amplifier is a tape recorder with automatic level control.)

A minimum set of compressed phonemes needs about 10 K bytes (for 40 phonemes, each occupying 256 bytes) of memory. Room is left over for BASIC programs or extra phonemes. With variable pitch and amplitude, you can accent syllables in words. Variable pitch plus extra long vowels could effectively make a singing Apple!■

Memory Locations Decimal	Hexadecimal	Usage
0-2047	000-07FF	System usage
2048-2303	0800-08FF	Blank
2304-2559	0900-09FF	Input subroutine
2560-2815	0A00-0AFF	Output subroutine
2816-LOMEM	0B00-LOMEM	Audio data storage
LOMEM-HIMEM		BASIC

Table 1: *Memory map for speech input and output routines.*

2a

Address Hexadecimal	Address Decimal	Variable Name	Suggested Value (Decimal)
0911	2321	THRESH	13
0915	2325	STARTLO	0
0916	2326	START HI	11
0923	2339	ENDHI	48 for 16 K bytes
092A	2346	ENDLO	0

2b

Address Hexadecimal	Address Decimal	Variable Name	Suggested Value (Decimal)
0A01	2561	STARTLO	0
0A02	2562	STARTHI	11
0A34	2612	DELAY	47

Table 2: *Tables of variable locations and values. Table 2a lists the location and suggested value of several constants that must be specified within listing 2; table 2b does the same for listing 3. In both cases, the constants are stored within the body of the listing.*

References

1. Chamberlin, Hal, "A Sampling of Techniques for the Computer Performance of Music," September 1977 BYTE, page 62.
2. Ciarcia, Steve, "Talk to Me," June 1978 BYTE, page 142.
3. Cross, Mark, "Apple Organ," a program based on reference 1.
4. O'Haver, Tom, "Audio Processing with a Microprocessor," June 1978 BYTE, page 166.